Prime Clocks

Michael Stephen Fiske

10th GI Conference on Autonomous Systems

October 23, 2017

AEMEA Institute

San Francisco, California

Motivation for Prime Clocks

- The mindset of mainstream computer science algorithms is SERIAL.

- Digital computers execute their machine instructions
  ONE INSTRUCTION (one step) at a TIME
  but there is no explicit reference to time.

- Active Element Machine (AEM), 2011.
  http://www.aemea.org/AEM

  (1) Every AEM command has an explicit reference to time.

  (2) Multiple AEM commands can execute at the same time.
  The AEM is an inherently parallel computing machine.

Digital Computers and Early Logic Gates

- Late 1930's, Digital Computers constructed from logic gates.[1,2]

- Based on Boolean logic[3], various forms of logic gates arose: mechanical switches, electro-mechanical devices, vacuum tubes.[4,5,6,7]

[1] Konrad Zuse. Patentanmeldung Z-2391, German Patent Office, 1941.

[2] Raul Rojas. Konrad Zuse's Legacy: The Architecture of the Z1 and Z3. IEEE Annals of the History of Computing, Vol. 19, No. 2, 1997.

[3] George Boole. An Investigation into the Laws of Thought. Walton and Maberly, 1854.

[4] H. Aiken & G. Hopper. The Automatic Sequence Controlled Calculator. Springer 203-222, 1982.

[5] Alice & Arthur Burks. The ENIAC. Annals of History of Computing, Vol. 3, No. 4, 310-399, 1981.

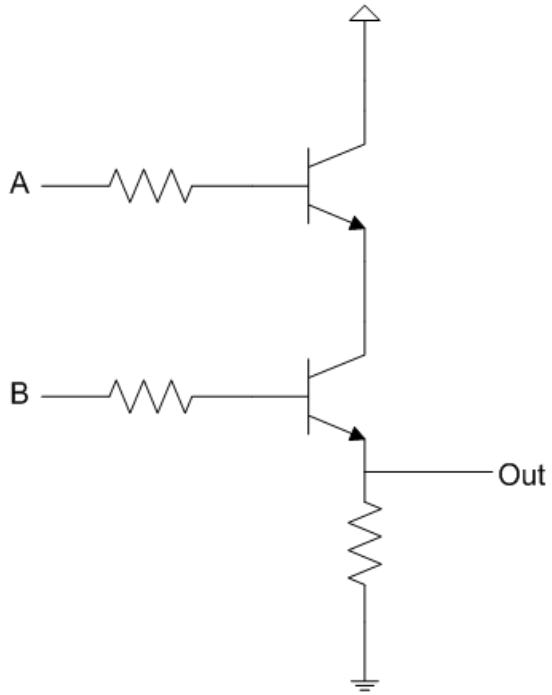
[6] C. Shannon. The synthesis of two-terminal switching circuits. Bell Systems Journal, 28, 59-98, 1949.

[7] Alan Turing. Proposals for Development in the Mathematics Division of ACE. Report E882. NPL, 1945

Transistors Transformed Logic Gates

- Transistors[8,9,10] transformed logic gates and enabled the integrated circuit[11,12].

AND gate



[8] J. Lilienfeld. Method and apparatus for controlling electric current. U.S. Patent 1,745,175. Oct. 8, 1926.

[9] J. Lilienfeld. Device for controlling electric current. U.S. Patent 1,900,018. Filed March 28, 1928.

[10] J. Bardeen & W.H. Brattain. The Transistor, A Semi-Conductor Triode. Physical Review, *74*, 230-231, June 25, 1948.

[11] Jack Kilby. Miniaturized Electronic Circuits. U.S. Patent 3,138,*743*, 1959.

[12] Robert N. Noyce. Semiconductor Device-and-Lead Structure. U.S. Patent 2,981,*877*, 1959.

AND, OR, NOT Logic Gates

- The conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ operators (i.e., AND, OR, NOT gates) form a Boolean algebra.

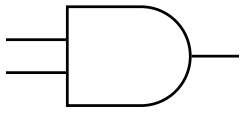
- Any Boolean function $f\colon \{0, 1\}^n \to \{0, 1\}$ can be constructed from a finite composition of AND, OR, NOT gates.

A	B	$A \wedge B$
0	0	0
1	0	0
0	1	0
1	1	1

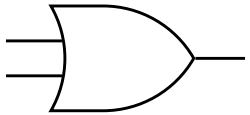
A	B	$A \vee B$
0	0	0
1	0	1
0	1	1
1	1	1

A	$\neg A$
0	1
1	0

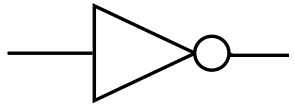
$A \wedge B$



$A \vee B$

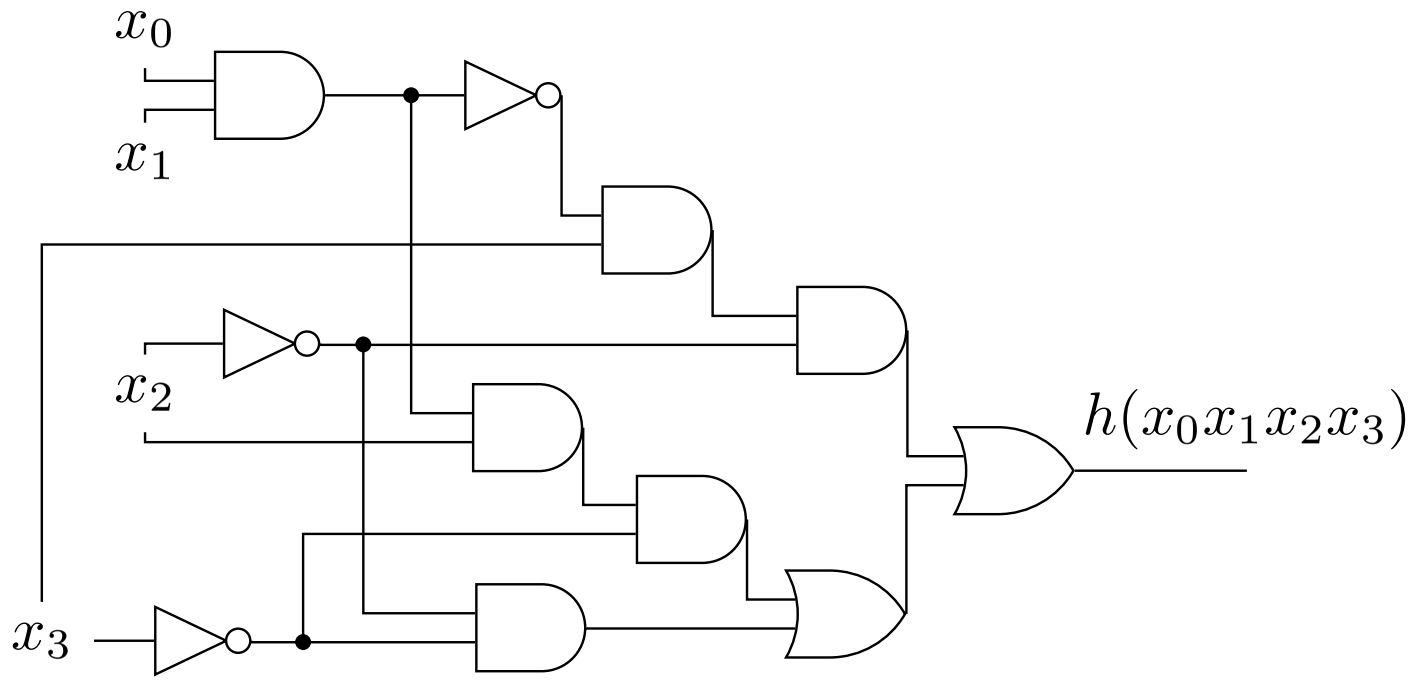


$\neg A$



Shannon's Theorem

- Most Boolean functions $f: \{0,1\}^n \to \{0,1\}$ built from AND, OR, NOT gates require about $\frac{2^n}{n}$ gates.



$$h(x_0x_1x_2x_3) = \big[\big(\neg(x_0 \wedge x_1)\big) \wedge (\neg x_2) \wedge x_3\big] \vee \big[x_0 \wedge x_1 \wedge x_2 \wedge (\neg x_3)\big] \vee \big[(\neg x_2) \wedge (\neg x_3)$$

Consequences of Boolean algebra and Shannon's Theorem

- Circuits built from AND, OR, NOT gates have a depth.

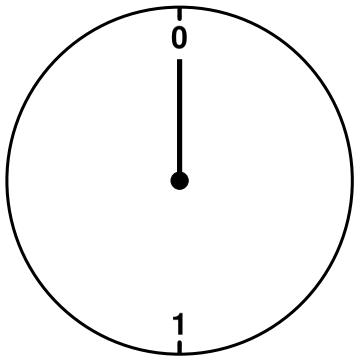
  $\neg(x \wedge y) \neq (\neg x) \wedge y$ because $\neg(0 \wedge 0) = 1$ and $(\neg 0) \wedge 0 = 0$.

Complete parallelism with logic gates is not attainable.

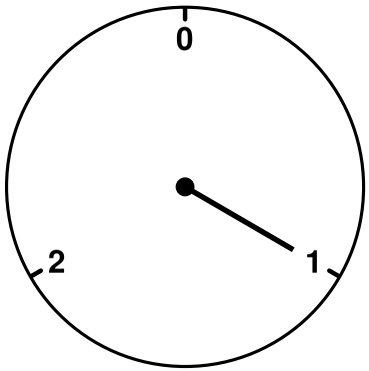
- The complexity of the graph connectivity of the gates for most functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is exponential as a function of n. Each gate has 1 or 2 input lines and 1 output line, so the exponential complexity follows from Shannon's theorem.

Prime Clocks

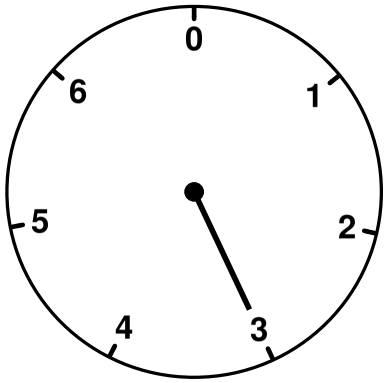
[2, 0] clock



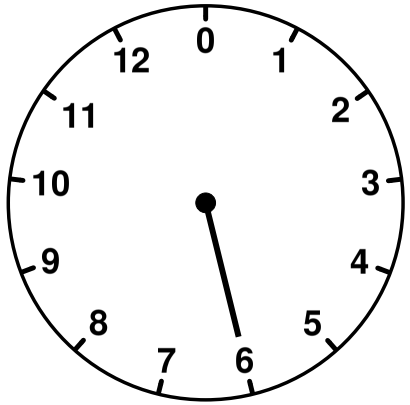
[3, 1] clock



[7, 3] clock



[13, 6] clock



A [2, 0] clock ticks 0, 1, 0, 1, 0, 1, . . .

A [3, 1] clock ticks 1, 2, 0, 1, 2, 0, . . .

A [7, 3] clock ticks 3, 4, 5, 6, 0, 1, 2, 3, 4, . . .

A [13, 6] clock ticks 6, 7, 8, 9, 10, 11, 12, 0, 1, 2, 3, 4, 5, 6, . . .

Prime Clock Sums in Ω_2

Time	$[2,0]$	$[3,1]$	$[2,0] \oplus [3,1]$	$[7,3]$	$[13,6]$	$[7,3] \oplus [13,6]$
0	0	1	**1**	3	6	**1**
1	1	2	**1**	4	7	**1**
2	0	0	**0**	5	8	**1**
3	1	1	**0**	6	9	**1**
4	0	2	**0**	0	10	**0**
5	1	0	**1**	1	11	**0**
...						

Ω_n is the set of all sequences $a : \mathbb{N} \to \{0, 1, \ldots n-1\}$

Prime clock sums in Ω_2 can compute Boolean functions.

For a and b in Ω_n, define $a +_n b = c$ where $c_i = (a_i + b_i) \bmod n$.

For $n \geq 2$, Ω_n with binary operator $+_n$ is a commutative group.

Commutativity means the clock computation can be parallelized.

A Prime Clock Sum in Ω_5

Time	$[5,3]$	$[7,6]$	$[11,3]$	$[13,0]$	$[5,3] \oplus_5 [7,6] \oplus_5 [11,3] \oplus_5 [13,0]$
0	3	6	3	0	2
1	4	0	4	1	4
2	0	1	5	2	3
3	1	2	6	3	2
4	2	3	7	4	1
5	3	4	8	5	0
6	4	5	9	6	4
7	0	6	10	7	3
8	1	0	0	8	4
...					

Notation: $\big([5,3] \oplus_5 [7,6] \oplus_5 [11,3] \oplus_5 [13,0]\big)(6) = 4$

Prime Clock sums can be computed in any Ω_n.

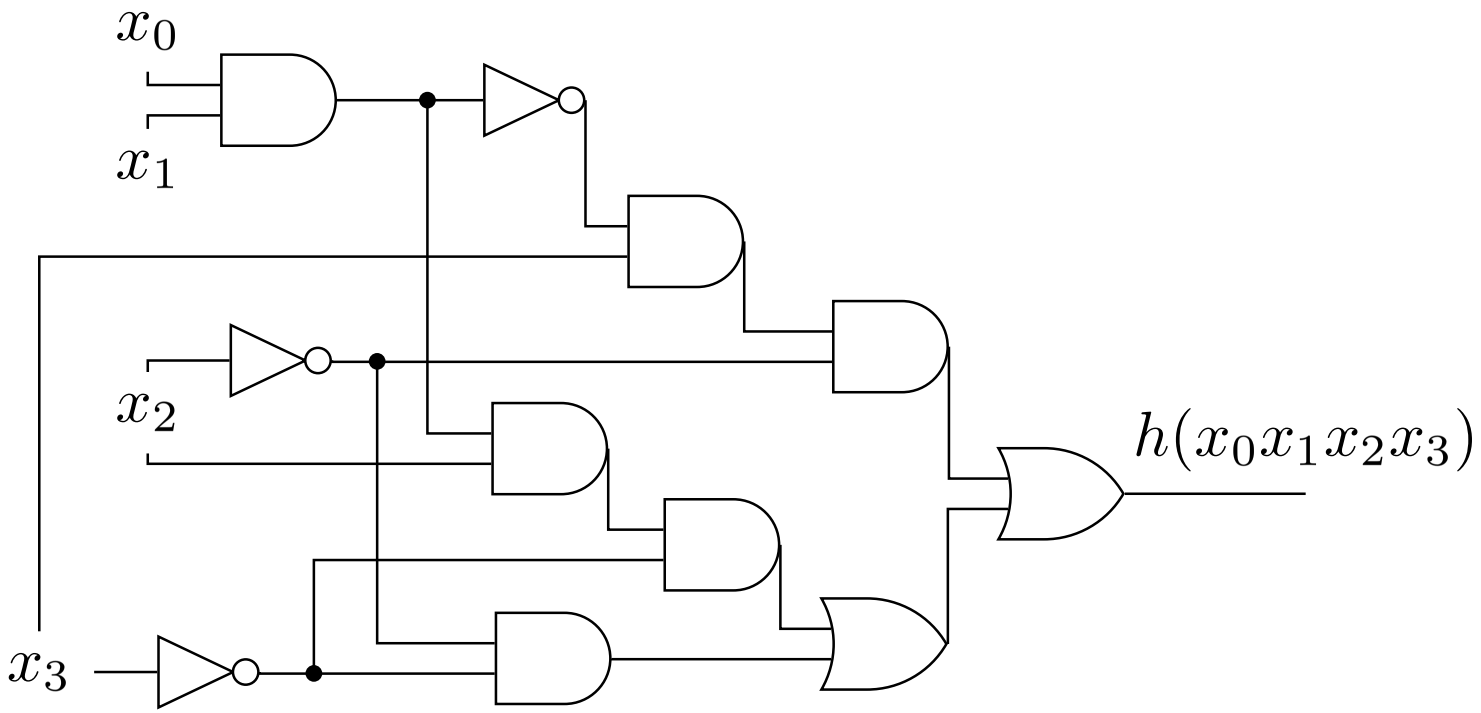
The rest of the talk will focus on prime clock sums in Ω_2.

Parallelization of Prime Clock Sums vs. Circuit Depth

With $m = x_0 + 2x_1 + 4x_2 + 8x_3$, prime clock sum $([7, 3] \oplus [13, 6])(m)$

computes the same function on $\{0, 1\}^4$ as function

$$h(x_0x_1x_2x_3) = \big[\big(\neg(x_0 \wedge x_1)\big) \wedge (\neg x_2) \wedge x_3\big] \vee \big[x_0 \wedge x_1 \wedge x_2 \wedge (\neg x_3)\big] \vee \big[(\neg x_2) \wedge (\neg x_3)$$



Prime Clock Sum Algorithm 1

Prime Clock Sum $[q_1, t_1] \oplus [q_2, t_2] \oplus \ldots \oplus [q_L, t_L]$

ALGORITHM 1.

INPUT: i

Set $b_1 = ((i + t_1) \bmod q_1) \bmod 2$

Set $b_2 = ((i + t_2) \bmod q_2) \bmod 2$

. . .

Set $b_L = ((i + t_L) \bmod q_L) \bmod 2$

Set $y = (b_1 + b_2 + \ldots + b_L) \bmod 2$

OUTPUT: y

Hardware computes $(b_1 + b_2 + \ldots + b_L) \bmod 2$ as a parallel exclusive-or.[13]

For $0 < k < L+1$, L distinct clocks can compute $((i + t_k) \bmod q_k) \bmod 2$ in paralle

[3] Hao Yan, Liping Feng, Thomas H. LaBean and John H. Reif. Parallel Molecular Computations of Pairwise Exclusive-Or (XOR) Using DNA String Tile Self-Assembly. J. Am. Chem. Soc., 125, 47, 14246–14247, 2003.

Prime Clock Sum Algorithm 2

Prime Clock Sum $[q_1, t_1] \oplus [q_2, t_2] \oplus \ldots \oplus [q_L, t_L]$

ALGORITHM 2.

INPUT: i

Set $b_1 = (i + t_1) \bmod q_1$

Set $b_2 = (i + t_2) \bmod q_2$

$\ldots$

Set $b_L = (i + t_L) \bmod q_L$

Set $y = (b_1 + b_2 + \ldots + b_L) \bmod 2$

OUTPUT: y

2 bit multiplication

$$M_0 = [2,0] \oplus [7,3] \oplus [7,4] \oplus [7,5] \oplus [11,10]$$

$$M_1 = [2,0] \oplus [2,1] \oplus [3,0] \oplus [5,2] \oplus [11,0] \oplus [11,1]$$

$$M_2 = [5,0] \oplus [7,0] \oplus [7,2] \oplus [11,4]$$

$$M_3 = [2,1] \oplus [5,0] \oplus [11,1] \oplus [11,6]$$

Multiplication functions $\mathcal{M}_i : \{0,1\}^4 \to \{0,1\}$.

u	l	$\mathcal{M}_3$	$\mathcal{M}_2$	$\mathcal{M}_1$	$\mathcal{M}_0$
00	00	0	0	0	0
00	01	0	0	0	0
00	10	0	0	0	0
00	11	0	0	0	0
01	00	0	0	0	0
01	01	0	0	0	1
01	10	0	0	1	0
01	11	0	0	1	1
10	00	0	0	0	0
10	01	0	0	1	0
10	10	0	1	0	0
10	11	0	1	1	0
11	00	0	0	0	0
11	01	0	0	1	1
11	10	0	1	1	0
11	11	1	0	0	1

Can the finite Prime Clock Sums compute all Boolean functions?

Yes. Showing this is true requires some work.

A group is a set G and binary operation + with 4 properties.

(I) + is closed on G.

(II) + is associative on G. $(a + b) + c = a + (b + c)$.

(III) G has a unique identity e. For all a in G, $a + e = e + a = a$.

(IV) Each a in G has a unique inverse g. $a + g = g + a = e$.

For positive integer m, define $B_m = \{ b_1 b_2 \ldots b_m : b_i \text{ is 0 or 1} \}$.

Set B_m has 2^m bit strings. Define binary operator $+_2$ on B_m as

$a_1 \ldots a_m +_2 b_1 \ldots b_m = c_1 \ldots c_m$ with $c_i = (a_i + b_i) \bmod 2$.

Set B_m with binary operation $+_2$ is a commutative group.

Prime clocks $[p, 0],\ [p, 1],\ \ldots,\ [p, p-1]$

Consider prime clocks $[p, 0],\ [p, 1],\ [p, 2],\ \ldots,\ [p,\ p-1]$.

Prime number p is called a 1 mod 4 prime if $(p-1)/2$ is even.

Prime number p is called a 3 mod 4 prime if $(p-1)/2$ is odd.

Prime clock sum $[q_1, t_1] \oplus [q_2, t_2] \oplus \cdots \oplus [q_L, t_L]$ is mapped to a bit string $c_0\, c_1 \ldots c_{p\text{-}1}$ in $B_p = \{0, 1\}^p$ where bit

$c_k = \left([q_1, t_1] \oplus [q_2, t_2] \oplus \cdots \oplus [q_L, t_L] \right)(k)$.

Theorem 3. If p is a 3 mod 4 prime, the subgroup of Ω_2 generated by clocks $[p, 0],\ [p, 1],\ \ldots\ [p, p-1]$ is isomorphic to $B_p = \{0, 1\}^p$

The subgroup generated by the 7 clocks

7 is a 3 mod 4 prime. The subgroup of Ω_2 generated by the clocks $[7,0]$, $[7,1]$, $[7,2]$, $[7,3]$, $[7,4]$, $[7,5]$, $[7,6]$ is isomorphic to $\{0,1\}^7$.

Time	$[7,0]$	$[7,1]$	$[7,2]$	$[7,3]$	$[7,0] \oplus [7,2] \oplus [7,3]$	$[7,1] \oplus [7,2] \oplus [7,3]$
0	0	1	0	1	1	0
1	1	0	1	0	0	1
2	0	1	0	1	1	0
3	1	0	1	0	0	1
4	0	1	0	0	0	1
5	1	0	0	1	0	1
6	0	0	1	0	1	1
...						

Theorem 5. If p is a 1 mod 4 prime, the subgroup of Ω_2 generated by clocks $[p,0]$, $[p,1]$, ... $[p,p-1]$ is isomorphic to B_{p-1}.

5 is a 1 mod 4 prime. In table 5, $[5,0]\oplus[5,1] = [5,2]\oplus[5,3]\oplus[5,4]$

Table 5: The 5-clocks projected into Ω_2

Time	$[5,0]$	$[5,1]$	$[5,2]$	$[5,3]$	$[5,4]$	$[5,0]\oplus[5,1]$	$[5,2]\oplus[5,3]\oplus[5,4]$
0	0	1	0	1	0	1	1
1	1	0	1	0	0	1	1
2	0	1	0	0	1	1	1
3	1	0	0	1	0	1	1
4	0	0	1	0	1	0	0
5	0	1	0	1	0	1	1
...							

Main Theorem.

For any positive integer n and any function $f: \{0, 1\}^n \rightarrow \{0, 1\}$, there exists a finite sum of prime clocks in Ω_2 that can compute f.

PROOF. Since there are an infinite number of primes, there is a prime number $p > 2^n$, so $(p - 1) \geq 2^n$. Now, p is a 1 mod 4 or a 3 mod 4 prime. Theorems 3, 5 imply that prime clocks $[p, 0]$, $[p, 1]$, . . . $[p, p - 1]$ generate a subgroup isomorphic to B_{p-1} or B_p. In either case B_{p-1} or B_p, for every bit string of length 2^n, there is a prime clock sum built from clocks $[p, 0]$, $[p, 1]$, . . . $[p, p - 1]$ that computes this bit string. One of these prime clock sums

computes the bit string:

$f(\,0^n\,)$

$f(\,0^{n-1}\,1\,)$

$f(\,0^{n-2}\,10\,)$

and so on all the way up to

$f(\,1^n\,)$

where

0^n corresponds to 0

$0^{n-1}\,1$ corresponds to 1

$0^{n-2}\,10$ corresponds to 2

and so on all the way up to

1^n corresponds to $2^n - 1$.

Summary

- When for each prime clock, there is one physical device, the computation is as parallel as a large XOR.

- Interesting: Parallel computation has been reduced to parallelizing a huge XOR computation.

- If a computational task can be transformed to one or more Boolean functions, then decentralized computation across a network is robust to timing and race conditions.

Questions - Part 1

- In some cases, using smaller clocks with different primes is more efficient than using clocks $[p, 0]$, $[p, 1]$, . . . $[p, p-1]$. Under what conditions are smaller clocks preferable?

- When one or more large primes are used, the mod 2 operation discards a lot of information. When the primes are large, can computation be made more efficient by working in Ω_n where $n >> 2$?

- What types of hardware or wetware (e.g., DNA computation) are suitable for building prime clocks and massively parallelizing XOR?

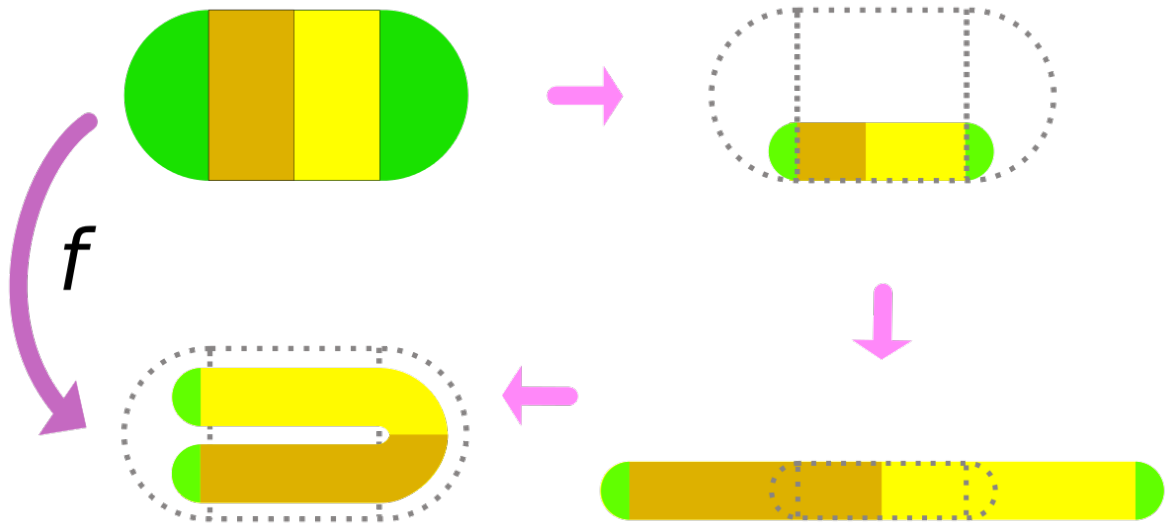
Questions - Part 2

- What are potential applications in machine learning and genetic algorithms?

- What are the potential applications in security and cryptography?

Questions - Strange Attractors

- In dynamical systems theory, the periodic points are often dense in the attractor and for each positive integer *n*, there is a periodic point with period *n*. Example: horseshoe map.[14]



- Do prime clocks provide any direction on how strange attractors can be used computationally?

[14] Stephen Smale. Differentiable Dynamical Systems. Bulletin of the American Mathematical Society. Vol. 73, No. 6, 747-817, 1967.